ACCNET—A corporate computer network

by MICHAEL L. COLEMAN

Aluminum Company of America
Pittsburgh, Pennsylvania

INTRODUCTION

The installation of a Digital Equipment Corporation DEC 10, in close proximity to an existing IBM 370/165, initiated an investigation into the techniques of supporting communication between the two machines. The method chosen, use a mini-computer as an interface, suggested the possibility of broadening the investigation into a study of computer networks—the linking of several large computer systems by means of interconnected mini-computers. This paper explains the concept of a network and gives examples of existing networks. It discusses the justifications for a corporate computer network, outlines a proposed stage by stage development, and analyzes and proposes solutions for several of the problems inherent in such a network. These include: software and hardware interfaces, movement of files between dissimilar machines, and file security.

WHAT IS A NETWORK?

A computer network is defined to be "an interconnected set of dependent or independent computer systems which communicate with each other in order to share certain resources such as programs or data—and/or for load sharing and reliability reasons."[19] In a university or a research environment, the network might consist of interconnected time-sharing computers with a design goal of providing efficient access to large CPUs by a user at a terminal. In a commercial environment a network would consist primarily of interconnected batch processing machines with a goal of efficiently processing a large number of programs on a production basis. One example of the use of a network in a commercial environment would be preparing a program deck on one computer, transmitting it to another computer for processing, and transmitting the results back to the first computer for output on a printer.

OTHER NETWORKS

Functioning networks have been in existence for several years.[4,19,36] These include: CYBERNET, a large commercial network consisting of interconnected Control Data Corporation machines;[38] the Distributed Computer System (DCS) at the University of California at Irvine;[18] the Michigan Educational Research Information Triad, Inc. (MERIT), a joint venture between Michigan State University, Wayne State University, and the University of Michigan;[2,12,30] the OCTOPUS System at the Lawrence Berkeley Laboratory;[41] the Triangle Universities Computation Center (TUCC) Network, a joint undertaking of the Duke, North Carolina State, and North Carolina Universities;[54] ad the TSS Network, consisting of interconnected IBM 360/67s.[39,47,53] But perhaps the most sophisticated network in existence today is the one created by the Advanced Research Projects Agency (ARPA), referred to as the ARPA network.[9,15,20,22,28,33,34,40,42,44,46] The ARPA network is designed to interconnect a number of various large time-shared computers (called Hosts) so that a user can access and run a program on a distant computer through a terminal connected to his local computer. It is set up as a message service where any computer can submit a message destined for another computer and be sure it will be delivered promptly and correctly. A conversation between two computers has messages going back and forth similar to the types of messages between a user console and a computer on a time-shared system. Each Host is connected to the network by a mini-computer called an Interface Message Processor (IMP). A message is passed from a Host to its IMP, then from IMP to IMP until it arrives at the IMP serving the distant Host who passes it to its Host. Reliability has been achieved by efficient error checking of each message and each message can be routed along two physically separate paths to protect against total line failures.

The ARPA network was designed to give an end-to-end transmission delay of less than half a second. Design estimates were that the average traffic between each pair of Hosts on the network would be .5 to 2 kilobits per second with a variation between 0 and 10 kilobits per second and the total traffic on the network would be between 200 and 800 kilobits per second for a 20 IMP network.[20] To handle this load, the IMPs were interconnected by leased 50KB lines.

For the initial configuration of the ARPA network, communication circuits cost $49,000 per node per year and the network supports an average traffic of 17 kilobits

From the collection of the Computer History Museum (www.computerhistory.org)

per node. Each IMP costs about $45,000 and the cost of the interface hardware is an additional $10,000 to $15,000.[23] The IMPs are ruggedized and are expected to have a mean time between failures of at least 10,000 hours—less than one failure per year. They have no mass storage devices and thus provide no long term message storage or message accounting. This results in lower cost, less down time, and greater throughput performance.[46]

TYPES OF NETWORKS

There are three major types of networks: Centralized, Distributed, and Mixed.[19]

A Centralized network is often called a "Star" network because the various machines are interconnected through a central unit. A network of this type either requires that the capabilities of the central unit far surpass those of the peripheral units or it requires that the central unit does little more than switch the various messages between the other units. The major disadvantage of a network of this type is the sensitivity of the network to failures in the central unit, i.e., whenever the central unit fails, no communication can occur. The most common example of this type of network is one consisting of a single CPU linked to several remote batch terminals.

A Distributed network has no "master" unit. Rather, the responsibility for communication is shared among the members; a message may pass through several members of the network before reaching its final destination. For reliability each unit in the network may be connected to at least two other units so that communication may continue on alternate paths if a line between two units is out. Even if an entire unit is disabled, unaffected members can continue to operate and, as long as an operable link remains, some communication can still occur. The ARPA network is an example of a Distributed network.

A Mixed network is basically a distributed network with attached remote processors (in most cases, batch terminals) providing network access to certain locations not needing the capability of an entire locally operated computer system. These remote locations are then dependent on the availability of various central CPUs in order to communicate with other locations.

Within a network, two types of message switching may occur: circuit switching and packet switching. Circuit switching is defined as a technique of establishing a complete path between two parties for as long as they wish to communicate and is comparable to the telephone network. Packet switching is breaking the communication into small messages or packets, attaching to each packet of information its source, destination, and identification, and sending each of these packets off independently to find its way to the destination. In circuit switching, all conflict and allocation of resources must be resolved before the circuit can be established thereby permitting the traffic to flow with no conflict. In packet switching, there is no dedication of resources and conflict resolution occurs during the actual flow. This may result in somewhat uneven delays being encountered by the traffic.[47]

WHY A NETWORK?

By examining the general characteristics of a network in the light of a corporate environment, specific capabilities which provide justification for the establishment of a corporate computer network can be itemized.[25] These are:

load balancing
avoidance of data duplication
avoidance of software duplication
increased flexibility
simplification of file backup
reduction of communication costs
ability to combine facilities
simplification of conversion to remote batch terminal
enhancement of file security

Load balancing

If a network has several similar machines among its members, load balancing may be achieved by running a particular program on the machine with the lightest load. This is especially useful for program testing, e.g., a COBOL compilation could be done on any IBM machine in the network and achieve identical results. Additionally, if duplicate copies of production software were maintained, programs could be run on various machines of the network depending on observed loads.

Avoidance of data duplication

In a network, it is possible to access data stored on one machine from a program executing on another machine. This avoids costly duplication of various files that would be used at various locations within the corporation.

Avoidance of software duplication

Executing programs on a remote CPU with data supplied from a local CPU may, in many cases, avoid costly software duplication on dissimilar machines. For example, a sophisticated mathematical programming system is in existence for the IBM 370. With a network, a user could conversationally create the input data on a DEC 10 and cause it to be executed on the 370. Without a network, the user would either have to use a more limited program, travel to the 370 site, or modify the system to run on his own computer.

Flexibility

Without a network each computer center in the corporation is forced to re-create all the software and data files it wishes to utilize. In many cases, this involves complete reprogramming of software or reformatting of the data files. This duplication is extremely costly and has led to considerable pressure for the use of identical hardware

From the collection of the Computer History Museum (www.computerhistory.org)

and software systems within the corporation. With a successful network, this problem is drastically reduced by allowing more flexibility in the choice of components for the system.

Simplification of file backup

In a network, file backup can be achieved automatically by causing the programs which update the file to create a duplicate record to be transmitted to a remote machine where they could be applied to a copy of the data base or stacked on a tape for batch update. This would eliminate the tedious procedure of manually transporting data from one machine to another; the resulting inherent delay in the updates would be eliminated.[11]

Reduction of communication costs

The substitution of a high bandwidth channel between two separate locations for several low bandwidth channels can, in certain cases, reduce communication costs significantly.

Ability to combine facilities

With a network, it is possible to combine the facilities found on different machines and achieve a system with more capability than the separate components have individually. For example, we could have efficient human interaction on one machine combined with a computational ability of a second machine combined with the capability of a third machine to handle massive data bases.

Simplification of conversion

Converting a site from its own computer to a remote batch terminal could be simplified by linking the computer at the site into the network during the conversion.

Enhancement of file security

By causing all references to files which are accessible from the network to go through a standard procedure, advanced file security at a higher level than is currently provided by existing operating systems may be achieved. This will allow controlled access to records at the element level rather than at the file level.

EXISTING SITUATION

The existing configuration of the DEC 10 installation provides a 300 (to be extended to 1200) baud link to the 370 via a COMTEN/60, a mini-computer based system which provides store-and-forward message switching capability for the corporate teletype network. This link is adequate to support the immediate needs of a Sales Order Entry System but is totally inadequate for the general capability of making the computational power and the massive file storage of the 370 available to a user on the DEC 10.

Five DATA 100 terminals provide remote batch service into the 370 for users at various locations including three plants and a research center. Most of the other plants have medium scale computer systems to support their local data processing needs. All make extensive use of process control mini-computers and two have UNIVAC 494 systems which can handle both real-time control and batch data processing.

Approximately 25 interactive CRTs scattered throughout various sales offices across the country have recently been installed to upgrade our Sales Order Entry System. Each terminal is connected to the DEC 10 on a dial-up 300 baud line.

PROPOSED SOLUTION

The most obvious solution to the problem of 370-DEC 10 communication would be to connect the DEC 10 to the 370 in a "back-to-back" fashion. To provide an upward flexibility, however, it is proposed that rather than connecting the machines in that way, they will be connected using a mini-computer as an interface. By designing the system which controls their interaction with a network approach, additional communication links may be obtained with a relatively small software investment. For example, if in the future, our research center obtains a large computer that they wish to incorporate into the communications process of the other two, an additional mini-computer would be placed there and connected via a communication line to the other.

This approach has several advantages. First, by going through a mini-computer, each of the two interfaces can be very carefully debugged in isolation and thus not affect the other machine. Second, once an IBM interface to the mini-computer is designed, one can connect any IBM machine into the network without rewriting any of the other interfaces. We would not have to write an IBM to UNIVAC interface, an IBM to CDC interface, an IBM to Honeywell interface, etc. Third, the only change necessary in the existing portion of the network, as the network expands, would be to inform the mini-computers of the presence of the other machines.

System description

In order to effectively describe a system as potentially complex as this one, we shall make use of techniques being developed under the classification of "Structured Programming."[17,37,48,55,56] The system will be broken down into various "levels of abstraction," each level being unaware of the existence of those above it, but being able to use the functions of lower levels to perform tasks and supply information. When a system is specified in terms

From the collection of the Computer History Museum (www.computerhistory.org)

of levels, a clear idea of the operation of the system may be obtained by examining each level, starting from the top, and continuing down until further detail becomes unimportant for the purposes of the specification.

Let us now examine the first few levels of a portion of the proposed system. The top-most level is level 6, under that is level 5, and so on. We shall look at what occurs in the case of a user at a terminal on the DEC 10 submitting a program to a distant IBM 370 under HASP.

- Level 6

On level 6 is found user and program processes. All interaction with the user or with a program written by the user occurs on this level. In fact, after this level is completely specified, the User Manual for the system can be written. In our example, an examination of what is happening would show the following steps:

User creates the input file and a file for the output;

User logs onto the network specifying his ID number;

User types "SUBMIT" command specifying the input file, the output file, and the Host on which the program is to be run. This submit command calls on the HASP Submit-Receive function on level 5;

User waits a brief period until he gets an "OK" from the terminal signifying that the program has been submitted. He is then free to either perform other actions or to sign off of the network;

At some later time the user receives an "output ready" message on his terminal;

User can now examine his output file.

- Level 5

On level 5 is found the HASP Submit-Receive function, HSR, and functions to perform network access control, file access control, and remote program control. Let us examine the actions of the HSR function applied to our example:

The HSR function obtains the name of the HASP-READER process of the specified Host. It then calls on a level 4 function to pass the input file to that process. When the level 4 function which controls process-to-process communication is completed, it will return a value corresponding to the job number that HASP has assigned;

The HSR function sends an "OK" to the user. It then obtains the name of the HASP-WRITER process on the specified Host and calls on a level 4 to pass the job number and to specify the output file to the HASP-WRITER. Control returns when the output file is complete;

The HSR function then sends an "OUTPUT READY" message to the user.

- Level 4

On level 4 is found the functions which control the file descriptors, file access, and process-to-process communication. Examining the actions of the process-to-process communication function, PPC, applied to our example, we find:

The PPC function converts the name of the process into a "well-known port" number and then establishes a logical link to the desired process;

It then formulates a message containing the information to be passed and uses a level 3 function to transmit the message;

It then receives a message in reply (which contains the job number in one case, and the output, in another). It passes this up to level 5 after destroying the links.

- Level 3

Level 3 contains, among others, the function which transfers a message from one Host to another. To do this it:

Takes the message, breaks it into pages, and calls a level 2 function to transmit each page;

When the last page has been transmitted, it waits for an acknowledgment;

If the acknowledgment indicates that a reply is being sent, it receives each page of the reply and passes up to level 4.

- Level 2

On level 2 is handled the passing of pages. The steps are:

The page is transferred from the Host to its IMP;

The page is then translated into the standard network representation and broken into packets;

From the collection of the Computer History Museum (www.computerhistory.org)

A level 1 function is called to transmit each packet.

- Level 1

At level 1 is handled the details of transmitting a packet from IMP to IMP. This includes retransmission in case of errors.

Stages of development

In order to allow the concept of a corporate computer network to be evaluated at minimum expense, it is desirable to break the development into discrete stages, each stage building on the hardware and software of the previous stage to add additional capability.

- Stage 1

This first stage would connect the DEC 10 to the local IBM 370/165 by using a single mini-computer. It would allow a user on the DEC 10 to conversationally build a program on a terminal and submit it to the 370 to be run under HASP. His output would be printed either at the 370, at the DEC 10, or at his terminal. This stage would also support the transfer of files consisting solely of character data to be transferred from one machine to the other.

The mini-computer hardware required for the stage would include: one CPU with 16-24K of memory, power monitor and restart, autoload, and teletype; two interfaces, one to the 370 and one to the DEC 10; a real time clock; and a cabinet. The approximate purchase price would be \$25,000 to \$35,000 with a monthly maintenance cost of approximately \$300. In addition, a disk and controller should be rented for program development. This cost is approximately \$500 per month and would be carried for the remaining stages.

- Stage 2

The second stage would remove the restriction on file transfer and allow files consisting of any type of data to be accessed from the other machine. At this stage, strict security controls would be integrated into the system.

The additional hardware required for this stage would include an additional CPU with 8K of memory and adaptors to interconnect the two CPUs. The approximate purchase cost would be \$9,000-\$12,000, with a monthly maintenance cost of approximately \$75.

- Stage 3

This stage would expand the network to include computers at other locations. Additional hardware at the original site would include one synchronous communication controller for each outgoing line at a cost of \$2,000-\$2,500 with a maintenance cost of \$25, and appropriate modems. Total cost for the original site, assuming two outgoing lines, would be between \$36,000 and \$49,500, excluding disk rental, modems, and communication lines.

- Stage 4

This stage could be developed in parallel with stage 3. It would add the capability for a user on a terminal attached to one machine to submit and interact with a program executing on the other machine. No additional hardware would be required.

- Stage 5

This stage consists of the design and implementation of automatic back-up procedures. Most of the preliminary analysis can be done in parallel with stages 2-4. These procedures would automatically create duplicate transactions of updates to critical files and have them routed to an alternate site to be applied to the back-up data base. No additional hardware is required.

HANDLING OF FILES IN A NETWORK

The handling of files in a non-homogeneous, distributed network poses several complex problems. These include control of access and transfer of information between dissimilar machines.

Control of access

That any system supporting multiple, simultaneous use of shared resources requires some sort of flexible, easy to use method of controlling access to those resources seems obvious to everyone (with the possible exception of the designers of IBM's OS/360), the main problem being how to provide the control at a reasonable cost. Restricting ourselves just to file access control, we see many potential methods with varying degrees of security and varying costs.[10,13,14,31,43] All provide control at the file level, some at the record level, and others at the element level. By designing our system with a Structured Programming approach, it should be possible to modify the method we choose, upgrading or downgrading the protection until a cost-benefit balance is reached.

Most designers of file access control systems have mentioned encryption of the data—we shall be no different. Apparently finding the cost prohibitive, they have failed to include this capability in their final product. In the proposed network, however, translation between the data representations of dissimiliar machines will be performed (see below), so the added cost of transforming from a "scrambled" to an "unscrambled" form will be small.

Each file access control system is based on a method which associates with each user-file pair a set of descriptors listing the rights or privileges granted to that user for that file (e.g., Read Access, Write Access, Transfer of Read Access to another user). Conceptualized as entries in a matrix, these descriptors are almost never stored as

From the collection of the Computer History Museum (www.computerhistory.org)

such due to its sparceness. Rather, they are stored as lists, either attached to each element of a list of users or attached to each element of a list of files.

Assuming that we have a system for controlling file access, one design question for a distributed network is where to store the file access descriptors? For example, let us look at a network with three machines: A, B, and C, and a file, F, located at A but created by a user at B. To be accessible from the other machines, the file must be known by them and therefore, each machine must have a file descriptor stating that file F is located at A. If we also distribute the file access descriptors, an unauthorized user at C could gain access to the file by obtaining control of his machine and modifying the file access descriptors. Hence, each file access descriptor should be stored at the same location as the file it protects.

Transfer of information

The complexity of transferring information between two machines is increased by an order of magnitude when dissimilar machines are involved.[1,7,8] Using ASCII as the standard network code allows the interchange of files containing character data but does not address the problem of different representations of numerical data, e.g., packed decimal, short floating point, long floating point, etc.

Two alternatives present themselves: either allow each machine to translate from the representation of every other machine to its own or use a standard network representation and have each machine translate between its own and the network's. The first is attractive when only a few different types of machines will be allowed on the network (If there are N different types of machines, then N(N-1) translation routines might have to be written). The second alternative requires more effort in developing the standard network representation, but is really the only choice when the number of different types is larger than three or four.

Another problem is the large amount of translation that must take place. It may not be desirable to place this CPU laden task on a time-sharing machine for fear of degrading response time so the solution seems to lie in executing the translation within the IMPs. If performing translation interferes with the ability of the IMP to perform communication, an additional CPU can be attached to each in order to perform this task. With hardware costs decreasing 50 percent every two or three years, this seems an attractive solution.

INTERFACES

IMP—Host interface

The ARPA network is optimized toward supporting terminal interaction.[28] A commercial network must be optimized toward maximizing throughput of lengthy data files which produces large peak loads requiring high bandwidth channels between each Host and its IMP. In order to allow an IMP to communicate with its Host with a minimum of CPU intervention by either party, data must be transferred directly between the memory of the IMP and the memory of the Host. This can be achieved by connecting to an equivalent of the memory bus of the DEC 10 or multiplexor channel of the 370. With this type of interconnection, it will be necessary to configure the software so that each member of the communicating partnership appears to the other member as if it were a peripheral device of some sort, presumably a high-speed tape drive. Communication, therefore, would take place by one member issuing a READ while the other member simultaneously issues a WRITE.[24]

IMP-IMP interface

The IMPs will be linked by standard synchronous communication interfaces. Initial plans call for 40.8KB full duplex leased lines, but 19.2KB lines could also be used. A Cyclical Redundancy Check will provide detection of errors and cause the offending packet to be retransmitted.

Software interfaces

One of the main reasons for using mini-computers between the Hosts is to insure that the number of interface programs which must be written only grows linearly with the number of different types of Hosts. The effort in writing subsequent versions of the IMP-Host interface can be minimized by at least two methods:

1. Put as much of the system software as possible into the IMPs. Make use of sophisticated architecture[3] (e.g., multi-processor mini-computers, read-only memory) to obtain the power required.
2. For that portion of the system which resides in the Host, write the software using a standard, high-level language (e.g., FORTRAN) for as much of the code as possible.

REFERENCES

1. Anderson, Robert, et al, *Status Report on Proposed Data Reconfiguration Services*, ARPA Network Information Center Document No. 6715, April 28, 1971.
2. Aupperle, Eric, "MERIT Computer Network: Hardware Considerations" *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 49-63.
3. Bell, G., Cady, R., McFarland, H., Delagi, B., O'Laughlin, J., Noonan, R., "A New Architecture for Mini-Computers—The DEC PDP-11," *Proc. AFIPS 1970 SJCC*, Vol. 36, AFIPS Press, Montvale, N.J., pp. 657-675.
4. Bell, G., Habermann, A. N., McCredie, J., Rutledge, R., Wulf, W., "Computer Networks," *Computer*, IEEE Computer Group, September/October, 1970, pp. 13-23.
5. Bjorner, Dines, "Finite State Automation—Definition of Data Communication Line Control Procedures," *Proc. AFIPS 1970 FJCC*, Vol. 37, AFIPS Press, Montvale, N.J., pp. 477-491.
6. Bowdon, Edward, K., Sr., "Network Computer Modeling" *Proc. ACM Annual Conference*, 1972, pp. 254-264.

From the collection of the Computer History Museum (www.computerhistory.org)

7. Bhushan, Abhay, *The File Transfer Protocol*, ARPA Network Information Center Document No. 10596, July 8, 1972.
8. Bhushan, Abhay, *Comments on the File Transfer Protocol*, ARPA Network Information Center Document No. 11357, August 18, 1972.
9. Carr, C. Stephen, Crocker, Stephen D., Cerf, Vinton G., "Host-Host Communication Protocol in the ARPA Network" *Proc. AFIPS 1970 SJCC*, Vol. 36, AFIPS Press, Montvale, N.J., pp. 589-597.
10. Carroll, John M., Martin, Robert, McHardy, Lorine, Moravec, Hans, "Multi-Dimensional Security Program for a Generalized Information Retrieval System," *Proc. AFIPS 1971 FJCC*, Vol. 39, AFIPS Press, Montvale, N.J., pp. 571-577.
11. Casey, R. G, "Allocation of Copies of a File in an Information Network," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 617-625.
12. Cocanower, Alfred, "MERIT Computer Network: Software Considerations," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 65-77.
13. Conway, R. W., Maxwell, W. L., Morgan, H. L., "On the Implementation of Security Measures in Information Systems" *Comm. of the ACM*, Vol. 15, April, 1972, pp. 211-220.
14. Conway, Richard, Maxwell, William, Morgan, Howard, "Selective Security Capabilities in ASAP—A File Management System," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 1181-1185.
15. Crocker, Stephen D., Heafner, John F., Metcalfe, Robert M., Postel, Jonathan B., "Function-Oriented Protocols for the ARPA Computer Network," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 271-279.
16. deMercado, John, "Minimum Cost-Reliable Computer Communication Networks," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 553-559.
17. Dijkstra, E. W., "The Structure of the 'THE' Multiprogramming System," *Comm. of the ACM*, Vol. 11, May, 1968.
18. Farber, David, "Data Ring Oriented Computer Networks" *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 79-93.
19. Farben, David J., "Networks: An Introduction," *Datamation*, April, 1972, pp. 36-39.
20. Frank, Howard, Kahn, Robert E., Kleinrock, Leonard, "Computer Communication Network Design—Experience with Theory and Practice," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 255-270.
21. Frank, Howard, "Optimal Design of Computer Networks," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 167-183.
22. Frank, H., Frisch, I.T., Chou, W., "Topological Considerations in the Design of the ARPA Computer Network," *Proc. AFIPS 1970 SJCC*, Vol. 36, AFIPS Press, Montvale, N.J., pp. 581-587.
23. Frank, Ronald A., "Commercial ARPA Concept Faces Many Roadblocks," *Computerworld*, November 1, 1972.
24. Fraser, A. G., "On the Interface Between Computers and Data Communications Systems," *Comm. of the ACM*, Vol. 15, July, 1972, pp. 566-573.
25. Grobstein, David L., Uhlig, Ronald P., "A Wholesale Retail Concept for Computer Network Management," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 889-898.
26. Hansen, Morris H., "Insuring Confidentiality of Individual Records in Data Storage and Retrieval for Statistical Purposes," *Proc. AFIPS 1971 FJCC*, Vol. 39, AFIPS Press, Montvale, N.J., pp. 579-585.
27. Hansler, E., McAuliffe, G. K., Wilkov, R. S., "Exact Calculation of Computer Network Reliability," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 49-54.
28. Heart, F. E., Kahn, R. E., Ornstein, S. M., Crowther, W. R., Walden, D. C., "The Interface Message Processor for the ARPA Computer Network," *Proc. AFIPS 1970 SJCC*, Vol. 37, AFIPS Press, Montvale, N.J., pp. 551-1567.
29. Hench, R. R., Foster, D. F., "Toward an Inclusive Information Network," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 1235-1241.
30. Herzog, Bert, "MERIT Computer Network," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 45-48.
31. Hoffman, Lance J., "The Formulary Model for Flexible Privacy and Access Controls," *Proc. AFIPS 1971 FJCC*, Vol. 39, AFIPS Press, Montvale, N.J., pp. 587-601.
32. Hootman, Joseph T., "The Computer Network as a Marketplace," *Datamation*, April, 1972, pp. 43-46.
33. Kahn, Robert, "Terminal Access to the ARPA Computer Network" *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 972, pp. 147-166.
34. Kleinrock, Leonard, "Analytic and Simulation Methods in Computer Network Design," *Proc. AFIPS 1970 SJCC*, Vol. 36, AFIPS Press, Montvale, N.J., pp. 569-579.
35. Kleinrock, Leonard, "Survey of Analytical Methods in Queueing Networks," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 185-205.
36. Lichtenberger, W., (Ed), *Tentative Specifications for a Network of Time-Shared Computers*, Document No. M-7, ARPA, September 9, 1966.
37. Liskov, B. H., "A Design Methodology for Reliable Software Systems," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 191-199.
38. Luther, W. J., "Conceptual Bases of CYBERNET," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 111-146.
39. McKay, Douglas B., Karp, Donald P., "IBM Computer Network/440," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 27-43.
40. McQuillan, J. M., Crowther, W. R., Cosell, B. P., Walden, D. C., Heart, F. E., "Improvements in the Design and Performance of the ARPA Network," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 741-754.
41. Mendicino, Samuel F., "OCTOPUS: The Lawrence Radiation Laboratory Network," *Computer Networks*, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 95-110.
42. Metcalfe, Robert M., "Strategies for Operating Systems in Computer Networks," *Proc. ACM Annual Conference*, 1972, pp. 278-281.
43. Needham, R. M., "Protection Systems and Protection Implementations," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 571-578.
44. Ornstein, S. M., Heart, F. E., Crowther, W. R., Rising, H. K., Russell, S. B., Michel, A., "The Terminal IMP for the ARPA Computer Network," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 243-254.
45. Roberts, Lawrence G., "Extensions of Packet Communication Technology to a Hand Held Personal Terminal," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 295-298.
46. Roberts, Lawrence G., Wessler, Barry D., "Computer Network Development to Achieve Resource Sharing," *Proc. AFIPS 1970 SJCC*, Vol. 36, AFIPS Press, Montvale, N.J., pp. 543-549.
47. Rutledge, Ronald M., Vareha, Albin L., Varian, Lee C., Weis, Allan H., Seroussi, Salomon F., Meyer, James W., Jaffe, Joan F., Angell, Mary Anne K., "An Interactive Network of Time-Sharing Computers," *Proc. ACM Annual Conference*, 1969. pp. 431-441.
48. Sevcik, K. C., Atwood, J. W., Grushcow, M. S., Holt, R. C., Horning, J. J., Tsichritzis, D., "Project SUE as a Learning Experience," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 331-338.
49. Stefferud, Einar, "Management's Role in Networking," *Datamation*, April, 1972, pp. 40-42.
50. Thomas, Robert H., Henderson, D., Austin, "McROSS—A Multi-Computer Programming System," *Proc. AFIPS 1972 SJCC*, Vol. 40, AFIPS Press, Montvale, N.J., pp. 281-293.

From the collection of the Computer History Museum (www.computerhistory.org)

51. Tobias, M. J., Booth, Grayce M., "The Future of Remote Information Processing Systems," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 1025-1035.
52. Walden, David C., "A System for Interprocess Communication in a Resource Sharing Computer Network," *Comm. of the ACM*, Vol. 15, April, 1972, pp. 221-230.
53. Weis, Allan H., "Distributed Network Activity at IBM," *Computer Networks*, R. Rustin (Ed.), Prentice Hall, Englewood Cliffs, N.J., 1972, pp. 1-25.
54. Williams, Leland H., "A Functioning Computer Network for Higher Education in North Carolina," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 899-904.
55. Wulf, William A., "Systems for Systems Implementors—Some Experience From BLISS," *Proc. AFIPS 1972 FJCC*, Vol. 41, AFIPS Press, Montvale, N.J., pp. 943-948.
56. Wulf, W. A., Russell, D. B., Habermann, A. N., "BLISS: A Language for Systems Programming," *Comm. of the ACM*, Vol. 14, December, 1971, pp. 780-790.

From the collection of the Computer History Museum (www.computerhistory.org)